CORRECTED VERSION*

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 13/00

A1

(11) International Publication Number: WO 00/29961

(43) International Publication Date: 25 May 2000 (25.05.00)

(21) International Application Number: PCT/US99/26901

(22) International Filing Date: 12 November 1999 (12.11.99)

(30) Priority Data:
09/191,291 13 November 1998 (13.11.98) US

(71) Applicant: SONICS, INC. [US/US]; Suite 620, 2440 West El Camino Real, Mountain View, CA 94040 (US).

(72) Inventors: WINGARD, Drew, Eric; 1050 Hewitt Drive, San Carlos, CA 94070 (US). ROSSEEL, Geert, Paul; 2181 Avy Avenue, Menlo Park, CA 94025 (US). TOMLINSON, Jay, S.; 5516 Del Oro Drive, San Jose, CA 95124 (US). ROBINSON, Lisa, A.; 16 Blackstone Drive, Boulder Creek, CA 95006 (US).

(74) Agents: SOBRINO, Maria, E. et al.; Blakely, Sokoloff, Taylor & Zafman, 7th Floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025–1026 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published

With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: COMMUNICATIONS SYSTEM AND METHOD WITH MULTILEVEL CONNECTION IDENTIFICATION

COMMUNICATIONS SYSTEM AND METHOD WITH MULTILEVEL CONNECTION IDENTIFICATION

FIELD OF THE INVENTION

The present invention relates to a communication system to couple computing sub-systems.

BACKGROUND OF THE INVENTION

Electronic computing and communications systems continue to include greater numbers of features and to increase in complexity. At the same time, electronic computing and communications systems decrease in physical size and cost per function. Rapid advances in semiconductor technology such as four-layer deep-sub-micron complimentary metal-oxide semiconductor (CMOS) technology, have enabled true "system-on-a-chip" designs. These complex designs may incorporate, for example, one or more processor cores, a digital signal processing (DSP) core, several communications interfaces, and graphics support in application-specific logic. In some systems, one or several of these extremely complex chips must communicate with each other and with other system components. Significant

has ordering constraints, it is necessary for the sub-system to identify each increment of data received or transmitted with a certain part of a certain data stream to distinguish between streams and to preserve order within a stream. This includes identifying a sub-system that is a source of a data transmission. Conventionally, such identification is limited to a non-configurable hardware identifier that is generated by a particular sub-system or component.

Current bus systems provide limited capability to preserve order in one transaction stream by supporting "split transactions" in which data from one transaction may be interleaved with data from another transaction in the same stream. In such a bus, data is tagged as belonging to one stream of data, so that it can be identified even if it arrives out of order. This requires the receiving sub-system to decode an arriving address to extract the identification information.

Current bus systems do not support true concurrency of operations for a sub-system that can process multiple streams of transactions over a single interconnect, such as a memory controller that handles access to a single dynamic random access memory (DRAM) for several clients of the DRAM. A DRAM controller may require information related to a source of an access request, a priority of an access request, ordering requirements, etc. Current communication systems do not provide for such information

The connection identifier is a an identifier of global scope that transfers information between interface modules or between functional blocks through their interface modules. Some functional blocks may require all the information provided by the connection identifier, while other functional blocks may require only the subset of information provided by the thread identifier.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of one embodiment of a complex electronics system according to the present invention.

Figure 2 is an embodiment of a system module.

Figure 3 is an embodiment of a system module.

Figure 4 is an embodiment of a communications bus.

Figure 5 is a timing diagram showing pipelined write transfers.

Figure 6 is a timing diagram showing rejection of a first pipelined write transfer and a successful second write transfer

Figure 7 is a timing diagram showing interleaving of pipelined read and write transfers.

Figure 8 is a timing diagram showing interleaved connections to a single target.

Figure 9 is a timing di

102 is a fully custom integrated circuit designed specifically to operate in system 100. Other embodiments may contain additional sub-systems of the same types as shown, or other types not shown. Other embodiments may also include fewer sub-systems than the sub-systems shown in system 100. Integrated circuit 102 includes sub-systems 102A, 102B, 102C, 102D and 102E. ASIC 104 includes functional blocks 101A, 104B and 104C. FPGA 110 includes functional blocks 110A and 110B. A functional block may be a particular block of logic that performs a particular function. A functional block may also be a memory component on an integrated circuit.

System 100 is an example of a system that may consist of one or more integrated circuits or chips. A functional block may be a logic block on an integrated circuit such as, for example, functional block 102E, or a functional block may also be an integrated circuit such as fully custom integrated circuit 102 that implements a single logic function.

Shared communications bus 112 provides a shared communications bus between sub-systems of system 100. Shared communication bus 114 provides a shared communications bus between sub-systems or functional blocks on a single integrated circuit. Some of the functional blocks shown are connected to interface

manages second level arbitration. Data flow block 806 includes data flow first-in first-out (FIFO) buffers between shared communications bus 814 and initiator functional block 816, in addition to control logic associated with managing a transaction between shared communications bus 814 and initiator functional block 816. The FIFO buffers stage both the address and data bits transferred between shared communications bus 814 and initiator functional block 816. In one embodiment, shared communications bus 814 implements a memory mapped protocol. Specific details of an underlying computer bus protocol are not significant to the invention, provided that the underlying computer bus protocol supports some operation concurrency. A preferred embodiment of a bus protocol for use with the present invention is one that supports retry transactions or split transactions, because these protocols provide a mechanism to deliver operation concurrency by interrupting a multi-cycle transaction to allow transfers belonging to other unrelated transactions to take place. These protocols allow for higher transfer efficiencies because independent transactions may use the bus while an initiator waits for a long latency target to return data that has been previously requested by an initiator.

Address/command decode block 808 decodes an address on shared

direct, high efficiency transport of data traffic between functional blocks on shared communications bus 1012. In one embodiment, shared communications bus 1012 supports a bus protocol that is a framed, time division multiplexed, fully pipelined, fixed latency communication protocol using separate address, data and connection identification wires. The bus protocol supports fine grained interleaving of transfers to enable high operation concurrency, and uses retry transactions to efficiently implement read transactions from target devices with long or variable latency. Details of the arbitration method used to access shared communications bus 1012 are not required to understand the present invention. The delay from when an initiator functional block drives the command and address until the target functional block drives the response is known as the latency of shared communications bus 1012. The bus protocol supports arbitration among many initiator functional blocks and target functional blocks for access to the bus. In the embodiment shown, arbitration for access to shared communications bus 1012 is performed by an initiator interface module, such as module 1004 of Figure 4. In other embodiments, arbitration is performed by functional blocks directly, or by a combination of interface modules and functional blocks. In one embodiment, a bus grant lasts for one pipelined bus cycle. The protocol does not forbid a single

deliver the requested read data in time, but promises to do so at a later time. In this case an initiator must reattempt the transfer at a later time.

Connection identifier (CONNID) lines 226 carry a multi-bit signal driven by an initiator bus owner to indicate which connection the current transfer is part of. A connection is a logical state, established by an initiator, in which data may pass between the initiator and an associated target. The CONNID typically transmits information including the identity of the functional block initiating the transfer and ordering information regarding an order in which the transfer must be processed. In one embodiment, the information conveyed by the CONNID includes information regarding the priority of the transfer with respect to other transfers. In one embodiment the CONNID is a eight-bit code. An initiator interface module sends a unique CONNID along with an initial address transfer of a connection. Later transfers associated with this connection (for example, data transfers) also provide the CONNID value so both sender and receiver (as well as any device monitoring transfers on shared communications bus 1012) can unambiguously identify transfers associated with the connection. One advantage of using a CONNID is that transfers belonging to different transactions can be interleaved arbitrarily between multiple devices on a per cycle basis. In one embodiment,

cycle 3 (a REQ-RESP latency later), initiator E drives write data DATAE0 on the DATA wires; simultaneously, the selected target A drives RESP wires 342 with the DVA code, indicating that A accepts the write data. By the end of cycle 3, target A has acquired the write data, and initiator E detects that target A was able to accept the write data; and the transfer has thus completed successfully.

Meanwhile (i.e. still in cycle 3), initiator E issues a pipelined WRITE transfer (address ADDRE1) to target A. The write data and target response for this transfer both occur on cycle 5, where the transfer completes successfully. Proper operation of many systems and sub-systems rely on the proper ordering of related transfers. Thus, proper system operation may require that the cycle 3 WRITE complete after the cycle 1 WRITE transfer. In Figure 6, the CONNID field conveys crucial information about the origin of the transfer that can be used to enforce proper ordering. A preferred embodiment of ordering restrictions is that the initiator and target collaborate to ensure proper ordering, even during pipelined transfers. This is important, because transfer pipelining reduces the total latency of a set of transfers (perhaps a single transaction), thus improving system performance (by reducing latency and increasing usable bandwidth).

According to the algorithm of one embodiment:

successfully, it is marked as retired and may be deleted from the queue. If the transfer does not complete successfully, it will typically be re-attempted, and thus can go back into arbitration for re-issue. If the transfer does not complete successfully, and it will not be re-attempted, then it should not be marked as retired until the next transfer, if it exists, is not marked as issued. This restriction prevents the initiator logic from issuing out of order. As the oldest non-Retired transfer issues, it is marked as issued. This allows the second-oldest non-retired transfer to arbitrate to issue until the older transfer completes (and is thus marked as non-issued), if it is marked as pipelined.

An embodiment of the target implementation maintains a time-ordered queue whose depth matches the REQ-RESP latency. The queue operates off of the bus clock, and the oldest entry in the queue is retired on each bus cycle; simultaneously, a new entry is added to the queue on each bus cycle. The CONNID from the current REQ phase is copied into the new queue entry. In addition, if the current REQ phase contains a valid transfer that selects the target (via the External Address), then "first" and "busy" fields in the new queue entry may be set; otherwise, the first and busy bits are cleared. The first bit will be set if the current transfer will receive a BUSY response (due to a resource conflict) and no earlier

priority to the CONNID values, and therefore decide which requests to act upon based upon a combination of the CONNID value and the internal state of the target. For instance, a target might have separate queues for storing transfer requests of different priorities. Referring to Figure 9, the target might have a queue for low priority requests (which present with an odd CONNID) and a queue for high priority requests (which present with an even CONNID). Thus, the CONNID 1 WRITE ADDRE0 request of cycle 1 would be rejected if the low-priority queue were full, whereas the CONNID 2 READ ADDRD0 transfer could be completed successfully based upon available high-priority queue resources. Such differences in transfer priorities are very common in highly-integrated electronic systems, and the ability for the target to deliver higher quality of service to higher priority transfer requests adds significantly to the overall predictability of the system.

As Figure 9 implies, the algorithm described above allows a target to actively satisfy transfer requests from multiple CONNID values at the same time. Thus, there may be multiple logical transactions in flight to and/or from the same target, provided that they have separate CONNID values. Thus, the present invention supports multiple connections per target functional block.

Additionally, an initiator may require the ability to present multiple

Signals shown in Figure 11 are labeled with signal names. In addition, some signal names are followed by a notation or notations in parentheses or brackets. The notations are as follows:

(I) The signal is optional and is independently configurable

(A) The signal must be configured together with signals having similar notations

(AI) The signal is independently configurable if (A) interface modules exist

[#] Maximum signal width

The clock signal is the clock of a connected functional block. The command (Cmd) signal indicates the type of transfer on the bus. Commands can be issued independent of data. The address (Addr) signal is typically an indication of a particular resource that an initiator functional block wishes to access. Request Accept (ReqAccept) is a handshake signal whereby slave 1104 allows master 1102 to release Cmd, Addr and DataOut from one transfer and reuse them for another transfer. If slave 1104 is busy and cannot participate in a requested transfer, master 1102 must continue to present Cmd, Addr and DataOut. DataOut is data sent from a master to a slave, typically in a write transfer. DataIn typically carries read data.

Response (Resp) and DataIn are signals sent from slave 1104 to master 1102. Resp indicates that a transfer request

a mechanism by which a system entity may associate particular transactions with the system entity. One use of the CONNID is in establishing request priority among various initiators. Another use is in associating actions or data transfers with initiator identity rather than the address presented with the transaction request.

The embodiment of Figure 11 provides end-to-end connection identification with CONNID as well as point-to-point, or more local identification with Thread ID. A Thread ID is an identifier of local scope that simply identifies transfers between the interface module and its connected functional block. In contrast, the CONNID is an identifier of global scope that identifies transfers between two interface modules (and, if required, their connected functional blocks).

A Thread ID should be small enough to directly index tables within the connected interface module and functional block. In contrast, there are usually more CONNIDs in a system than any one interface module is prepared to simultaneously accept. Using a CONNID in place of a Thread ID requires expensive matching logic in the interface module to associate a returned CONNID with specific requests or buffer entries.

Using a networking analogy, the Thread ID is a level-2 (data link layer) concept, whereas the CONNID is more like a level-3 (network layer)

A DMA a engine is an example of an initiator functional block that also functions as a target functional block. When the DMA engine is programmed by software, it acts as a target. Thereafter, the DMA engine is an initiator. Because a DMA engine performs both read and write operations, two connections can be associated with a single DMA engine. If some buffering is available in the DMA engine, read and write operations may be decoupled so that both types of operations can be performed concurrently. A read may occur from a long latency storage device which requires the read data to be buffered on the DMA engine before a write operation writes the data. In one embodiment, each of DMA engines 1202 uses a Thread ID to identify the read stream and a different Thread ID to identify the write stream. The DMA engine does not require more information, such as what other functional block participates in a transaction. Therefore, a CONNID is not required to be sent from the DMA engine 1202 to a connected interface module 1204. Mapping of a Thread ID to a CONNID occurs in the interface module 1204.

In one embodiment, each initiator interface module 1204 maps a unique CONNID to each of two Thread Ids from a connected DMA engine 1202. Each of DMA engines 1202 use a single bit, for example, Thread ID of Figure 11, to

IN THE CLAIMS

What is claimed is:

1. A communication system comprising:

at least two functional blocks, wherein an first functional block communicates with a second functional block by establishing a connection, wherein a connection is a logical state in which data may pass between the first functional block and the second functional block; and

a bus coupled to each of the functional blocks and configured to carry a plurality of signals, wherein the plurality of signals comprises a connection identifier that indicates a particular connection that a data transfer is part of.

2. The communication system of claim 1, wherein the plurality of signals further comprises a thread identifier that indicates a transaction stream that the data transfer is part of.

3. The communication system of claim 2, further comprising:

a request thread busy signal that indicates that indicates to an initiator functional block that the target functional block cannot receive new requests associated with certain threads; and

a response thread busy signal that indicates that the initiator functional block cannot receive any new responses from the target functional block that are associated with certain threads.

10. A method for communicating between functional blocks in a computer system, the method comprising the steps of:

establishing a plurality of connection identifiers, wherein each connection identifier associates a particular data transfer with a particular connection, wherein a connection is a logical state in which data may pass between an initiator functional block of the plurality of functional blocks and a target functional block of the plurality of functional blocks, and wherein a connection is established when a particular data transfer is initiated; and

allowing an initiator functional block to issue a first transfer "Y" if the transfer "Y" is an oldest, non-issued, non-retired transfer among a set of transfer requests with a same connection identifier as the transfer "Y".

20. The method of claim 19, further comprising the step of, if the transfer is successfully completed, marking the transfer as retired; and deleting the transfer from the queue.

21. The method of claim 20, further comprising the step of, if the transfer is not successfully completed, re-attempting the transfer.

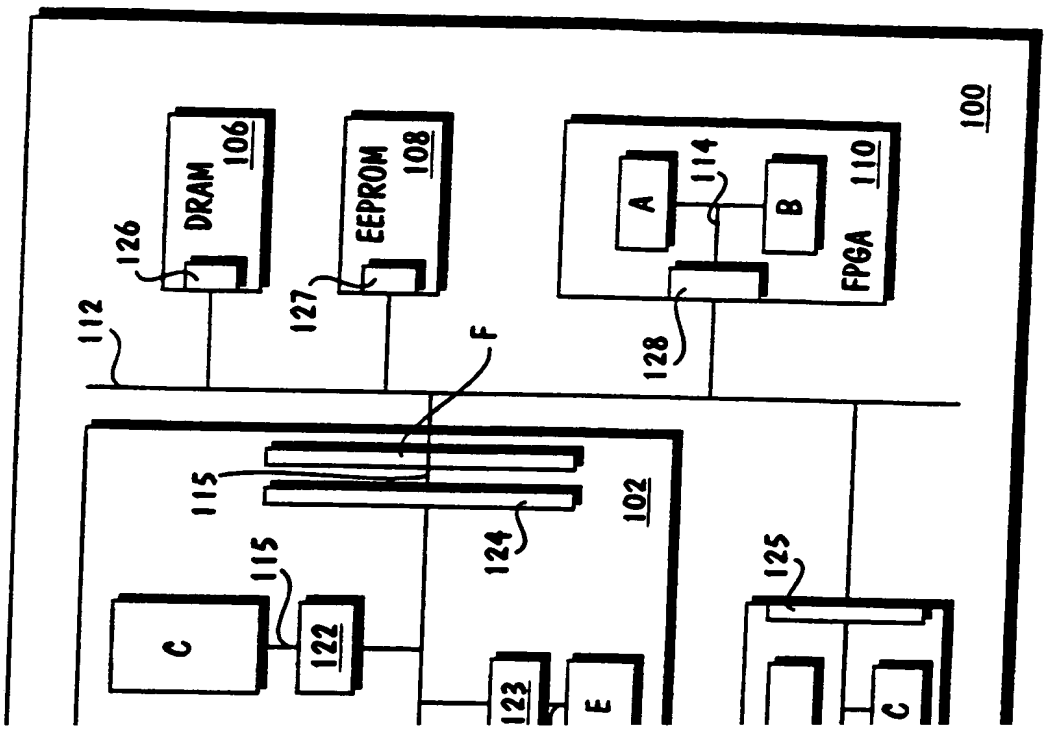
22. The method of claim 14, further comprising the step of the target functional block maintaining a time-ordered queue having a depth that is a number of bus clock cycles between a request for a transfer and a response to the request.

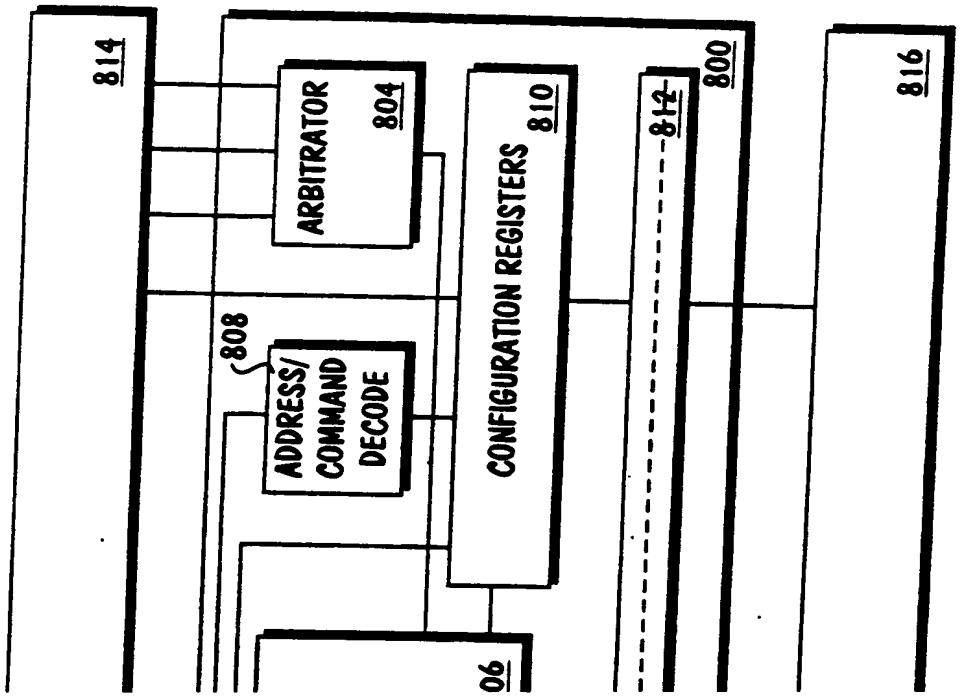
23. The method of claim 22, further comprising the steps of:

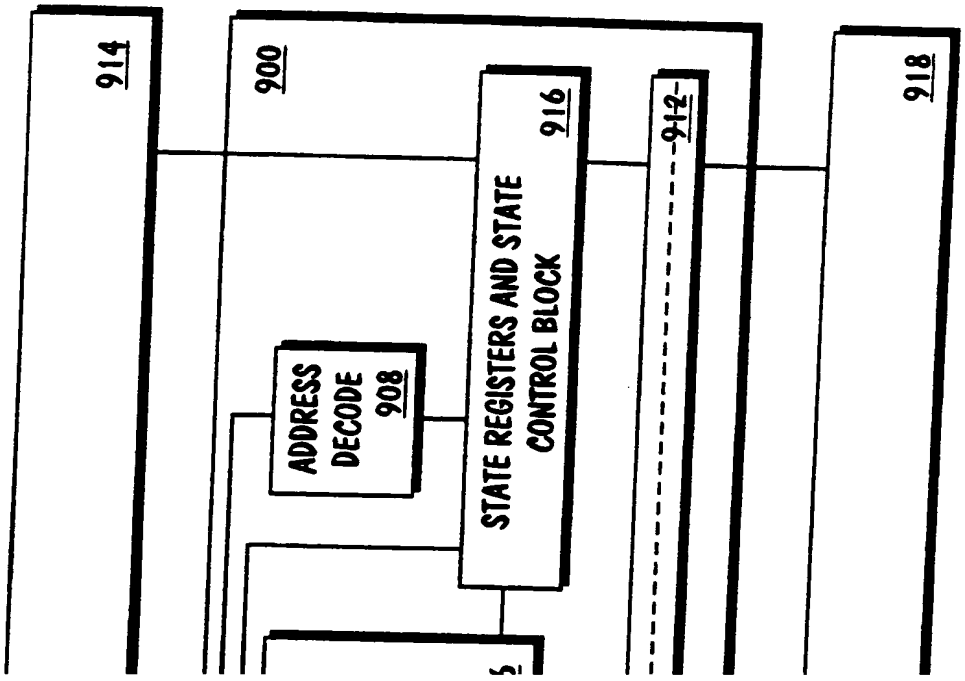
on each cycle of the bus clock, retiring an oldest entry in the time-ordered queue; and

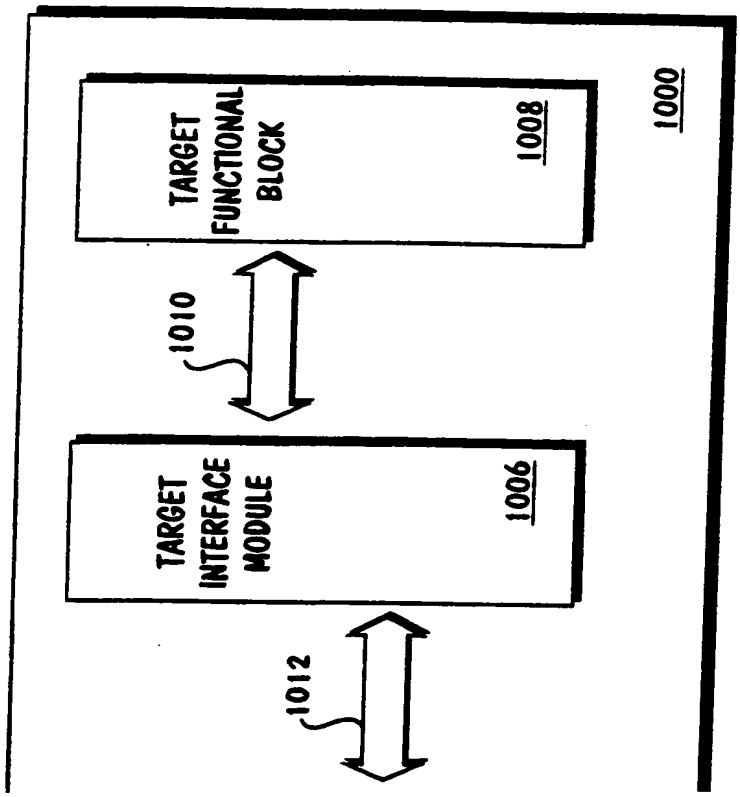
on each cycle of the bus clock, adding a new entry to the time-ordered queue, including a connection identifier associated with a current request for a transfer.

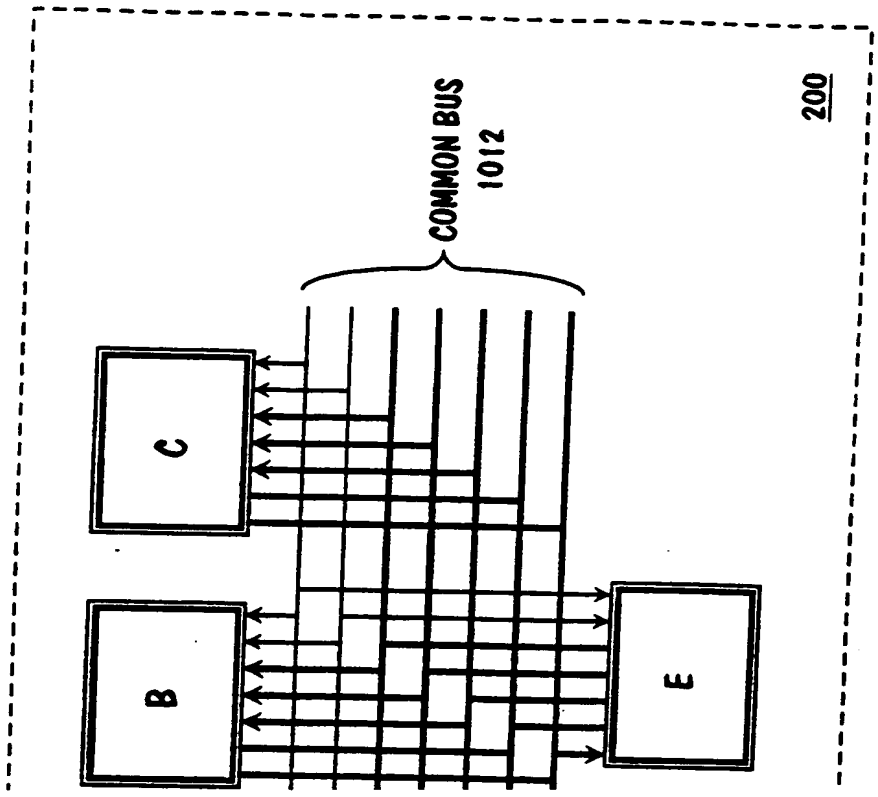
24. The method of claim 23, further comprising the steps of:

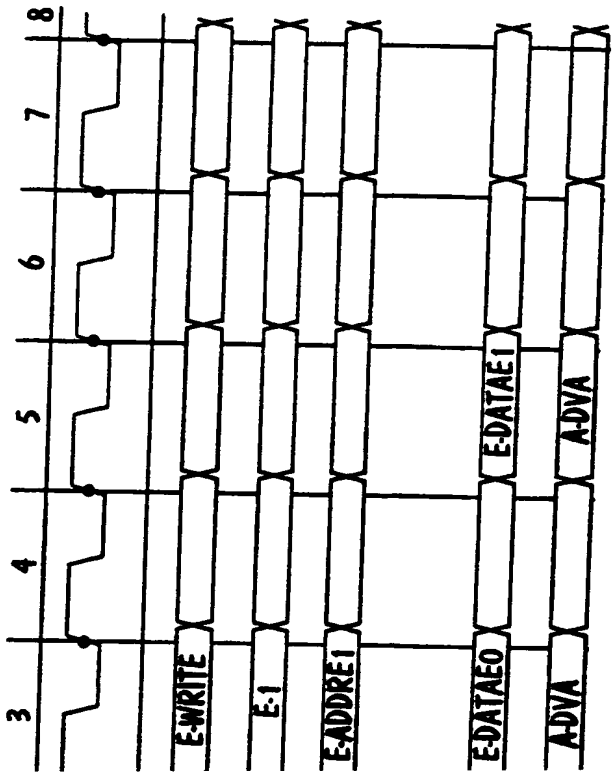

100
DRAM 106
EEPROM 108
FPGA 110
A
B
114
126
127
128
112
F
115
102
124
125
C
122
123
E
C

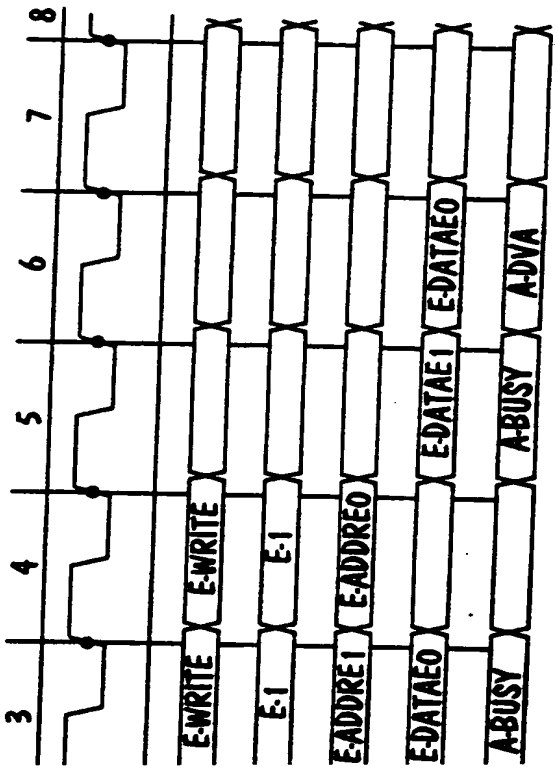

814
ARBITRATOR
804
808
ADDRESS/
COMMAND
DECODE
CONFIGURATION REGISTERS
810
812
800
816
06

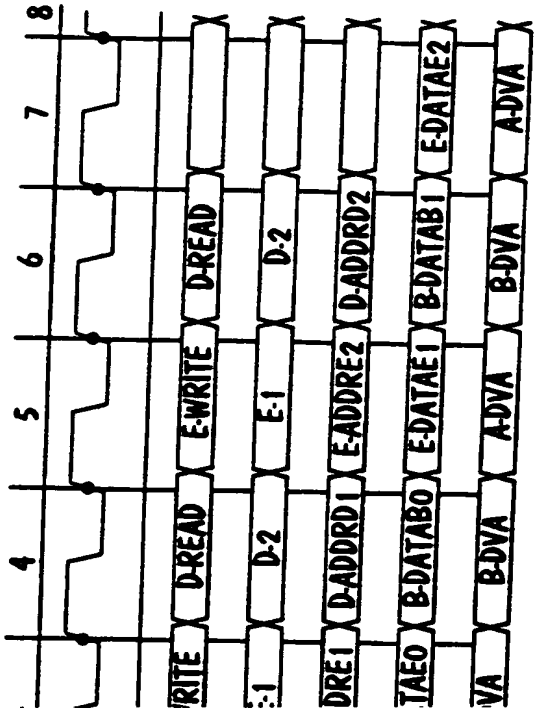

914
900
916
STATE REGISTERS AND STATE CONTROL BLOCK
912
918
ADDRESS DECODE
908

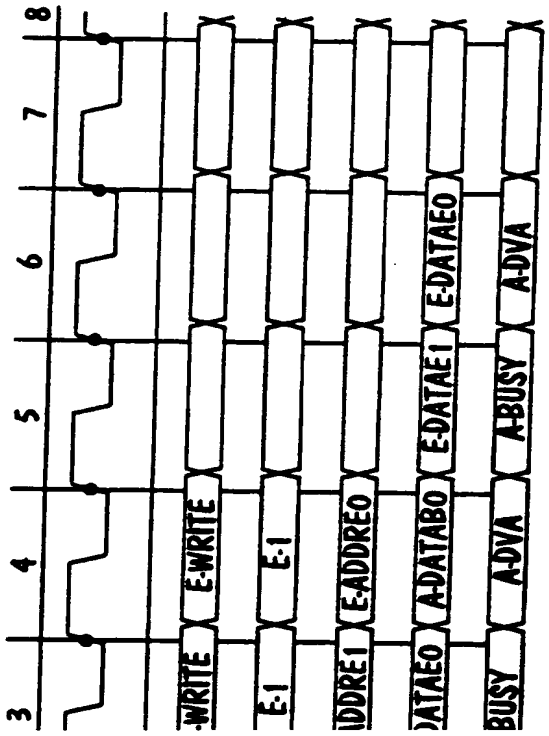

1000
TARGET FUNCTIONAL BLOCK
1008
1010
TARGET INTERFACE MODULE
1006
1012

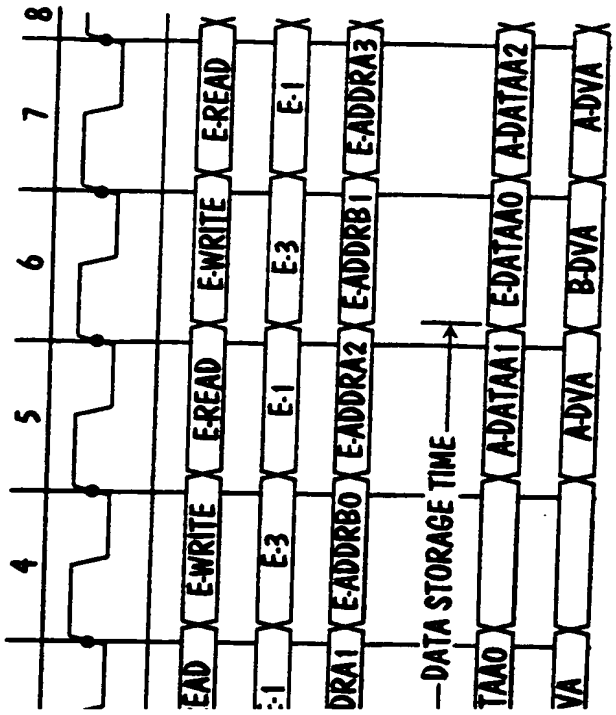

200
COMMON BUS
1012
C
B
E

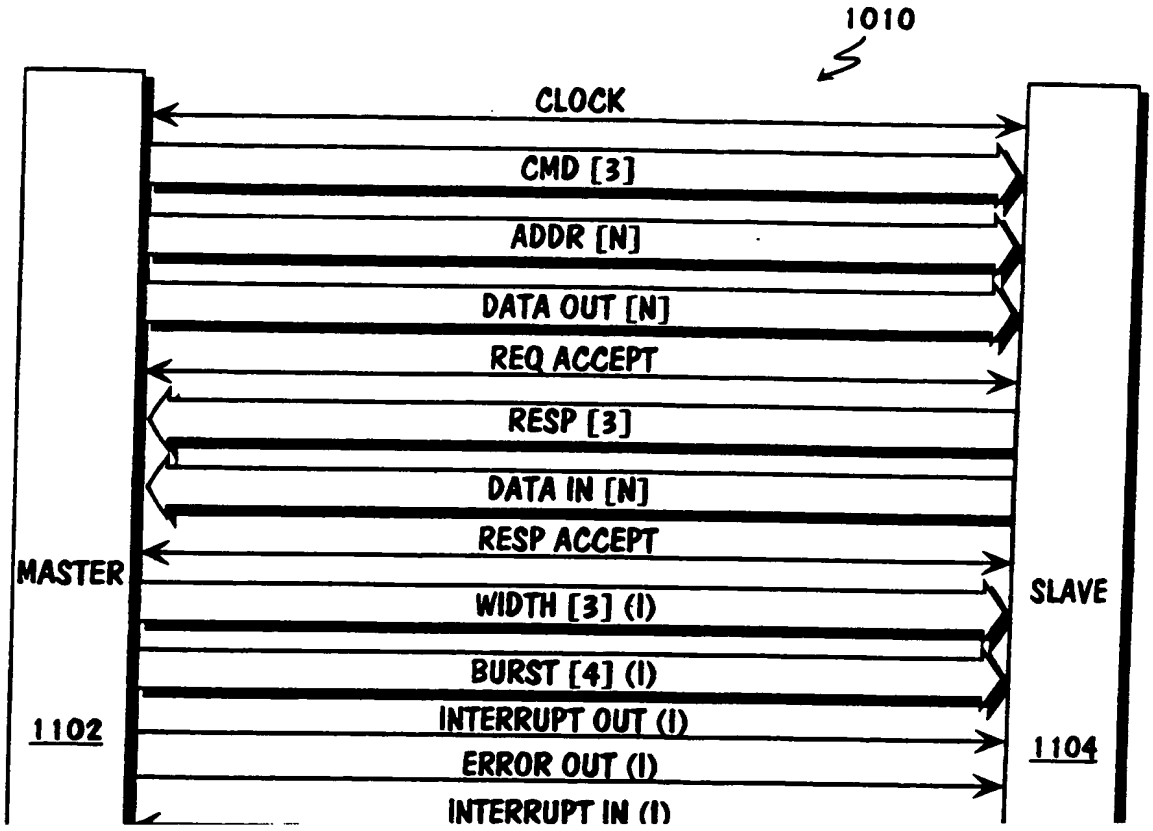

8
7
6
5
4
3
E-WRITE
E-1
E-ADDRE1
E-DATAE0
E-DATAE1
A-DVA
A-DVA

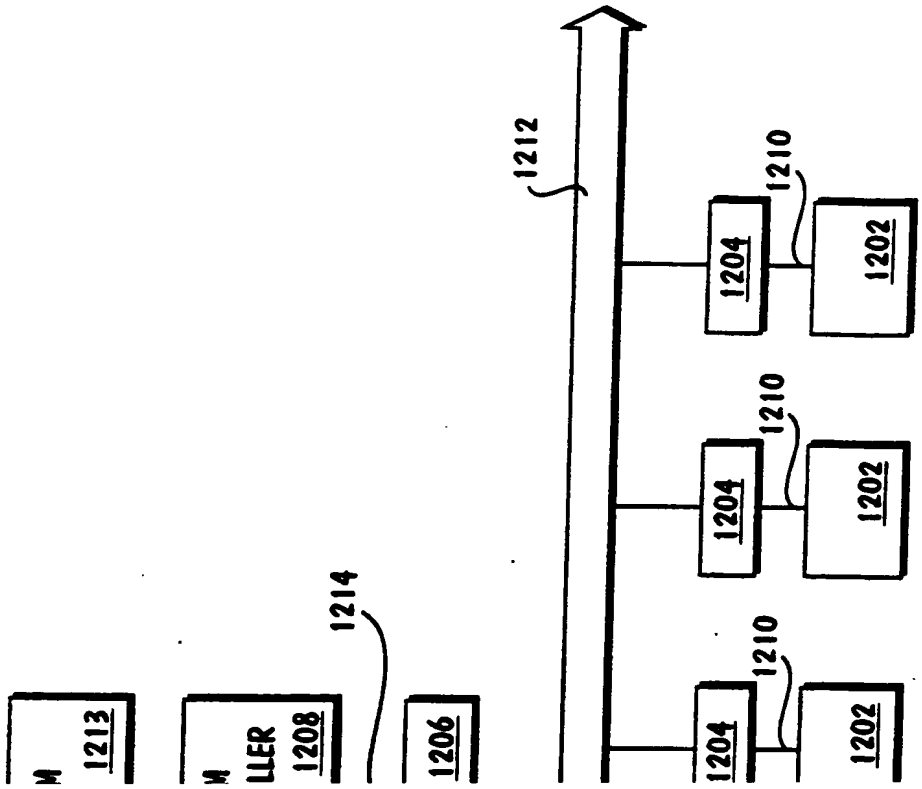

8
7
6
5
4
3
E-WRITE
E-1
E-ADDRE0
E-DATAE1
E-DATAE0
A-DVA
A-BUSY
E-WRITE
E-1
E-ADDRE1
E-DATAE0
A-BUSY


8
7
6
5
4
RITE
D-READ
E-WRITE
D-READ
:-1
D-2
E-1
D-2
DRE1
D-ADDRD1
E-ADDRE2
D-ADDRD2
TAE0
B-DATAB0
E-DATAE1
B-DATAB1
E-DATAE2
VA
B-DVA
A-DVA
B-DVA
A-DVA


8
7
6
5
4
3
E-WRITE
E-1
E-ADDRE0
E-DATAE1
E-DATAE0
A-DVA
A-BUSY
A-DVA
A-DATABO
E-1
WRITE
DDRE1
DATAE0
BUSY


8
7
6
5
4
E-READ
E-WRITE
E-READ
E-WRITE
EAD
E-1
E-3
E-1
E-3
-1
E-ADDRA3
E-ADDRB1
E-ADDRA2
E-ADDRBO
DRA1
DATA STORAGE TIME
A-DATAA2
E-DATAAO
A-DATAA1
TAAO
A-DVA
B-DVA
A-DVA
VA


1010
CLOCK
CMD [3]
ADDR [N]
DATA OUT [N]
REQ ACCEPT
RESP [3]
DATA IN [N]
RESP ACCEPT
WIDTH [3] (I)
BURST [4] (I)
INTERRUPT OUT (I)
ERROR OUT (I)
INTERRUPT IN (I)
MASTER
SLAVE
1102
1104


1212
1210
1204
1202
1210
1204
1202
1210
1204
1202
1214
1206
LLER 1208
1213

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/26901

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 13/00
US CL :710/100

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 710/100, 101, 106, 110, 126, 129, 1, 9 and 268

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

None

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

PLUS and STN

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,748,914 A (BARTH et al) 05 May 1998, cols. 2, 6, 10 and 11.	1-28
X	US 5,274,783 A (HOUSE et al) 28 December 1993, abstract and claims	1-28

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.